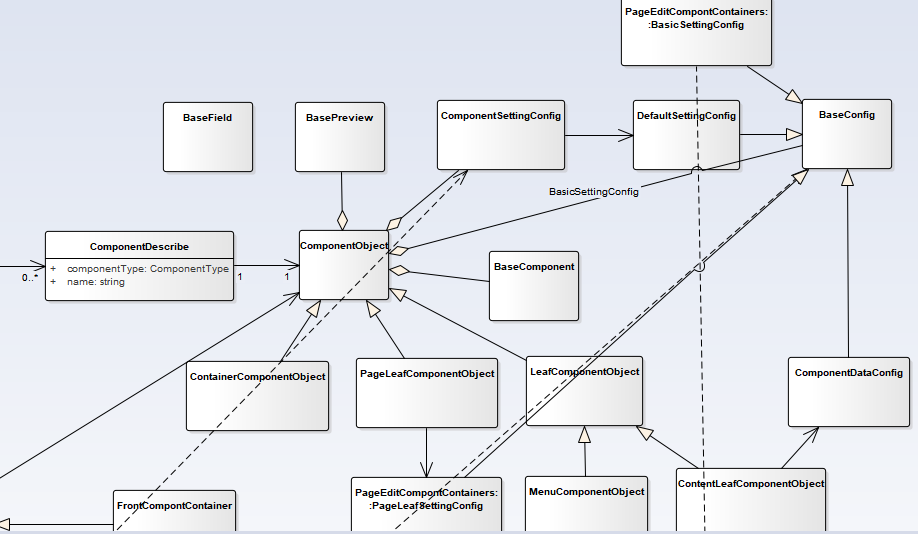
**组件设计图**



**概念**

**ComponentDescribe（组件描述）：**

每个开发的组件都应该默认导出一个ComponentDescribe，组件描述包含一个组件对象

**ComponentObject（组件对象）：**

组件对象封装了一系列我们需要的对象，其包含了BasePreview（预览），ComponentSettingConfig（组件设置配置），BaseComponent（组件）

**BaseComponent（基本组件）：**

组件是我们真正显示的内容

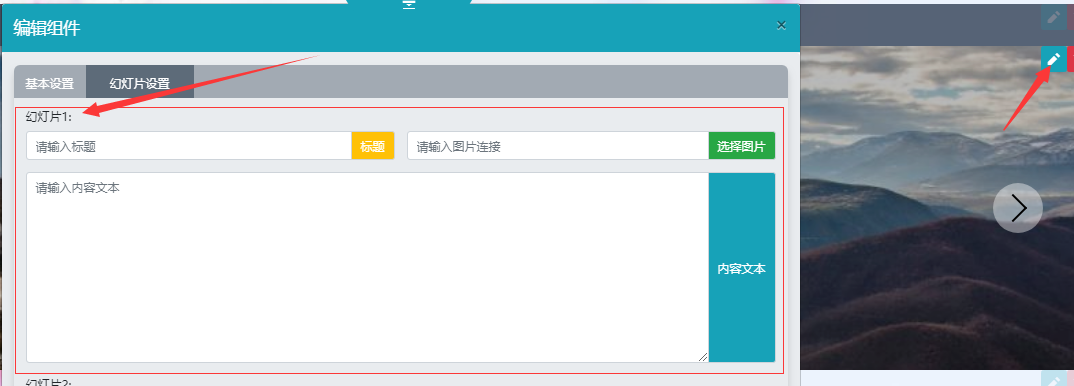
**BasePreview（预览）：**

预览是在组件预览框中显示的内容，如图：



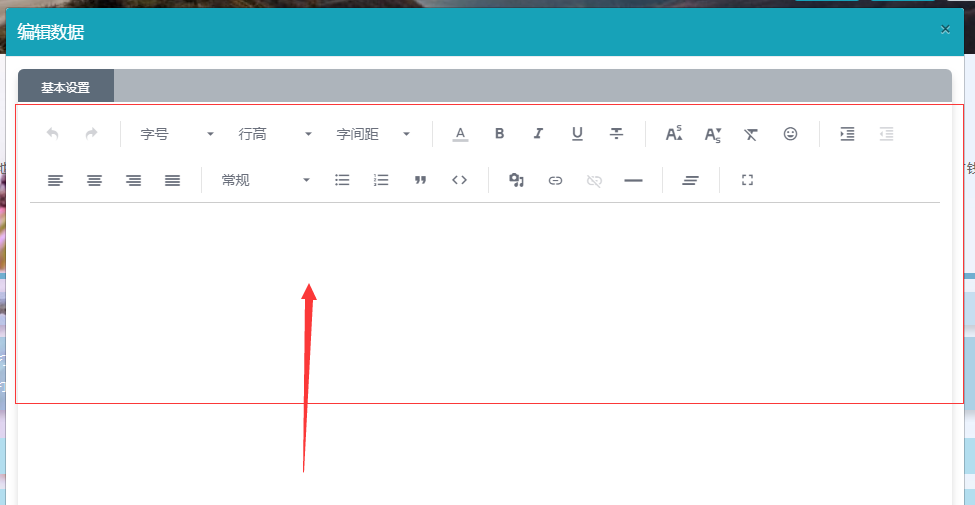
**ComponentSettingConfig（组件设置配置）：**

组件设置配置是编辑页面时设置中显示的内容，如图：



**ComponentDataConfig（组件数据配置）：**

组件数据配置是编辑数据时显示的内容。如图：



**组件的类型**

IE实现了几个基本组件类型，所有开发的组件必须继承其中一个基本组件

**基本容器组件**

容器组件可以包含其他组件

基本容器组件位于BaseContainerComponent，其导出ContainerComponentObject（容器组件对象）类和BaseContainerComponent（基本容器组件）类

**基本页叶子组件**

页叶子组件具有其指向的页面的文章数据

基本页叶子组件位于BasePageLeafComponent，其导出PageLeafComponentObject和BasePageLeafComponent

**基本静态组件**

静态组件什么功能都没有，只有展示效果

基本静态组件位于BaseStaticComponent，其导出StaticComponentObject和BaseStaticComponent

**基本菜单组件**

菜单组件具有菜单数据

基本菜单组件位于BaseMenuComponent，其导出MenuComponentObject和BaseMenuComponent

**基本内容叶子组件**

内容叶子组件具有文章数据

基本内容叶子组件位于BaseContentLeafComponent，其导出ContentLeafComponentObject和BaseContentLeafComponent，ComponentDataConfig（组件数据配置）

**组件开发**

**容器组件**

以Container组件为例：

组件代码：

import React from 'react'

import {BaseContainerComponent} from '../BaseContainerComponent'

import './Container.css'

// 基本容器组件类

class Container extends BaseContainerComponent

{

    constructor(props){

        super(props);

}

    render(){

        // 显示子组件

        return (<div className="containercss">{this.props.children}</div>);

    }

}

// 导出组件

export default Container;

组件对象代码：

import React from 'react'

import BaseComponentObject from '../BaseContainerComponent'

import Container from './Container'

import ComponentDescribe, {componentType} from '../ComponentDescribe'

// 继承基本容器组件对象

class ComponentObject extends BaseComponentObject {

    // 实现组件方法

Component(props, childrens) {

// 返回组件

        return <Container {...props} >{childrens}</Container>

    }

    // 实现预览方法

    Preview() {

        return <p>容器组件</p>;

    }

}

// 导出组件描述

let componentDescribe = new ComponentDescribe("Container", new ComponentObject(), componentType.container);

export default componentDescribe;

**静态组件**

以FilpClock为例

组件代码：

import React from 'react'

import {BaseStaticComponent} from '../BaseStaticComponent'

import './FlipClock.css'

// 进程基本静态组件

class FlipClock extends BaseStaticComponent

{

...

}

// 导出组件

export default FlipClock;

组件对象代码：

import React from 'react'

import BaseComponentObject from '../BaseStaticComponent'

import FlipClock from './FlipClock.jsx'

import ComponentDescribe, {componentType} from '../ComponentDescribe'

// 继承基本静态组件对象

class ComponentObject extends BaseComponentObject {

    constructor(){

        super();

    }

    // 实现组件方法

    Component(props) {

        return <FlipClock {...props} />

    }

    // 实现预览方法

    Preview() {

        return <p>翻页式时钟</p>;

    }

}

// 导出组件描述

let componentDescribe = new ComponentDescribe("FlipClock", new ComponentObject(), componentType.other);

export default componentDescribe;

**页叶子组件**

BasePageLeafComponent提供了page（PageModel结构）和pageDatas（PageDataModel结构）数据，和一个默认数据方法

class BasePageLeafComponent extends BaseComponent

{

    constructor(props) {

        super(props);

        this.state = {

            page: {},

            pageDatas: []

        }

}

    // 获取页面文章数据，如果没有，则返回基本组件提供的示例数据

    getPageDatasOrDemoDatas(){

        if(this.state.pageDatas.length == 0){

            return [{

                "id": 1,

                "name": "post1",

                "title": "圣战系谱"

            },

            {

                "id": 2,

                "name": "post2",

                "title": "圣魔之光石"

            },

            {

                "id": 3,

                "name": "post3",

                "title": "苍炎之轨迹"

            },

            {

                "id": 4,

                "name": "post4",

                "title": "晓之女神"

            },

            {

                "id": 5,

                "name": "post5",

                "title": "风花雪月"

            }];

        }

        return this.state.pageDatas;

    }

...

以PostList为例

组件代码：

import React from 'react'

import { BasePageLeafComponent } from '../BasePageLeafComponent'

import { NavLink } from 'react-router-dom'

import './Index.css';

// 继承基本页叶子组件

export default class PostList extends BasePageLeafComponent {

    constructor(props) {

        super(props)

}

    render() {

        let list = this.getPageDatasOrDemoDatas().map((item, index) => (

            <li key={item.id}>

                <span><span>{index + 1}</span></span>

                <NavLink className="text-secondary" to={`/Page/${this.state.page.name}/${item.name}`}>{item.title}</NavLink>

            </li>

        ));

        return (

            <div className="postlist">

                <ol className="postlist-rounded-list">

                    {list}

                </ol>

            </div>

        );

    }

}

组件对象代码：

import React from 'react'

import PostList from './PostList'

import BaseComponentObject from '../BasePageLeafComponent';

import ComponentDescribe, {componentType} from '../ComponentDescribe'

// 继承基本页叶子组件对象

class ComponentObject extends BaseComponentObject {

    Component(props) {

        return <PostList {...props} />

    }

    Preview() {

        return <p>文章列表</p>;

    }

}

let componentDescribe = new ComponentDescribe("PostList", new ComponentObject(), componentType.page);

export default componentDescribe;

**菜单组件**

基本菜单组件提供了menus数据（MenuModel结构）和createUrl方法

export default class BaseMenuComponent extends BaseComponent {

    constructor(props) {

        super(props);

        this.state = {

            menus: []

        };

}

    // 生成 url

    createUrl(menu)

    {

        ...

        return url;

    }

}

以IEMenu为例

组件代码：

import React from 'react';

import { NavLink } from 'react-router-dom';

import {BaseMenuComponent} from '../BaseMenuComponent'

import './IEMenu.css'

// 继承基本组件

export default class IEMenu extends BaseMenuComponent{

    createRootMenu(menu, menuIndex)

    {

        ...

}

    render() {

        return (

            <div className='ie-menu'>

                {this.state.menus.map((item, index) => this.createRootMenu(item, index))}

            </div>

        );

    }

}

组件对象代码：

import React from 'react'

import BaseComponentObject from '../BaseMenuComponent'

import IEMenu from './IEMenu'

import ComponentDescribe, {componentType} from '../ComponentDescribe'

// 继承基本组件对象

class ComponentObject extends BaseComponentObject{

    Component(props) {

        return <IEMenu {...props} />

    }

    Preview(){

        return <p>IE菜单</p>;

    }

}

const componentDescribe = new ComponentDescribe("IEMenu", new ComponentObject(), componentType.menu);

export default componentDescribe;

**内容叶子组件**

内容叶子组件的props提供了componentData数据（ContentComponentDataModel结构）

以text为例

组件代码：

import React from 'react'

import {BaseContentLeafComponent} from '../BaseContentLeafComponent'

// 继承基本组件

class Text extends BaseContentLeafComponent

{

    getComponentData(){

        return this.props.componentData || {};

}

    render()

    {

        let text = this.getComponentData().field1 || "这是一段文本";

        return (<p>{text}</p>);

    }

}

export default Text;

组件对象代码：

import React from 'react'

import {BaseField} from '../BaseComponent';

import BaseComponentObject, { ComponentDataConfig } from '../BaseContentLeafComponent'

import Text from './Text'

import ComponentDescribe, {componentType} from '../ComponentDescribe';

const field1 = (props) => (<BaseField text={"文本"} {...props} />);

// 继承基本组件对象

class ComponentObject extends BaseComponentObject {

    constructor(){

        super();

        // 组件数据配置

        this.ComponentDataConfig = (props) => (

            <ComponentDataConfig {...props}

                // 指定field1使用的配置

                field1={field1}

            />);

    }

    Component(props) {

        return <Text {...props} />

    }

    Preview() {

        return <p>文本框</p>;

    }

}

let componentDescribe = new ComponentDescribe("Text", new ComponentObject(), componentType.text);

export default componentDescribe;

**配置**

配置是一个react组件，配置的props如下：

BaseConfig.propType = {

    data: PropTypes.object,

    setData: PropTypes.func.isRequired,

}

IE提供的默认配置

DefaultSettingConfig：

默认设置配置，其props如下：

DefaultSettingConfig.propTypes = {

    data: PropTypes.object,

    setData: PropTypes.func.isRequired,

    // 组件 props={fieldValue, setFieldValue}

    field1: PropTypes.func,

    field2: PropTypes.func,

    field3: PropTypes.func,

    field4: PropTypes.func,

    field5: PropTypes.func,

}

ComponentDataConfig：

默认组件数据配置，其props如下：

ComponentDataConfig.propTypes = {

    data: PropTypes.object,

    setData: PropTypes.func.isRequired,

    // 组件

    field1: PropTypes.func,

    field2: PropTypes.func,

    field3: PropTypes.func,

    field4: PropTypes.func,

    field5: PropTypes.func,

}

**设置配置**

每个组件都可以进行设置配置

在组件对象中添加设置配置：

import React from 'react';

import BaseComponentObject from '../BaseStaticComponent';

import { ComponentSettingConfig, DefaultSettingConfig } from '../BaseComponent';

import CitiesSlider from './CitiesSlider.jsx';

import ComponentDescribe, { componentType } from '../ComponentDescribe';

import CustomizeField from './CustomizeField';

const field1 = (props) => (<CustomizeField text={"幻灯片1"} {...props} />);

const field2 = (props) => (<CustomizeField text={"幻灯片2"} {...props} />);

const field3 = (props) => (<CustomizeField text={"幻灯片3"} {...props} />);

class ComponentObject extends BaseComponentObject {

    constructor() {

        super();

        // 指定设置配置

        this.ComponentSettingConfigs = [

            // 添加一个幻灯片设置

            new ComponentSettingConfig("pic", "幻灯片设置",

                // 指定生成配置的方法

                // pageComponentSetting 为 PageComponentSettingModel 结构

                // setPageComponentSetting 设置组件设置，结构为 (pageComponentSetting) => void

                (pageComponentSetting, setPageComponentSetting) => {

                    // 使用默认设置配置

                    return (<DefaultSettingConfig

                        data={pageComponentSetting}

                        setData={setPageComponentSetting}

                        // 指定字段1使用的配置

                        field1={field1}

                        field2={field2}

                        field3={field3}

                    />);

                }

            )];

    }

    Component(props) {

        return <CitiesSlider {...props} />

    }

    Preview() {

        return <p>幻灯片</p>;

    }

}

let componentDescribe = new ComponentDescribe("CitiesSlider", new ComponentObject(), componentType.other);

export default componentDescribe;

在组件中使用设置：

组件的props包含pageComponentSettings数据（PageComponentSettingModel结构）

updateSliders(props){

    ...

    // 查找 pic 配置

    let pageComponentSetting = props.pageComponentSettings.find(item=>item.name == "pic");

    // 配置是否有添加

    if(!pageComponentSetting){

        ...

        return;

    }

    // 查看字段1是否有数据

    if(pageComponentSetting.field1){

        // 使用字段1的数据

        sliders.push(JSON.parse(pageComponentSetting.field1));

}

    if(pageComponentSetting.field2){

        sliders.push(JSON.parse(pageComponentSetting.field2));

}

    if(pageComponentSetting.field3){

        sliders.push(JSON.parse(pageComponentSetting.field3));

}

    ...

}

**数据配置**

自定义数据配置

import React from 'react'

import {BaseField} from '../BaseComponent';

import BaseComponentObject, { ComponentDataConfig } from '../BaseContentLeafComponent'

import Text from './Text'

import ComponentDescribe, {componentType} from '../ComponentDescribe';

const field1 = (props) => (<BaseField text={"文本"} {...props} />);

class ComponentObject extends BaseComponentObject {

    constructor(){

        super();

        // 指定数据配置

        this.ComponentDataConfig = (props) => (

            // 使用默认的数据配置

            <ComponentDataConfig {...props}

                // 指定字段1配置

                field1={field1}

            />);

    }

    Component(props) {

        return <Text {...props} />

    }

    Preview() {

        return <p>文本框</p>;

    }

}

let componentDescribe = new ComponentDescribe("Text", new ComponentObject(), componentType.text);

export default componentDescribe;

使用数据：

import React from 'react'

import PropTypes from 'prop-types'

import {BaseContentLeafComponent} from '../BaseContentLeafComponent'

class Text extends BaseContentLeafComponent

{

    getComponentData(){

        return this.props.componentData || {};

}

    render()

    {

        // 从componentData中获取字段1的数据

        let text = this.getComponentData().field1 || "这是一段文本";

        return (<p>{text}</p>);

    }

}

export default Text;

**字段**

前面讲的配置有应用字段，字段是一个react组件

字段的props如下，其接收一个字段值和一个设置字段值的方法：

BaseField.propTypes = {

    fieldValue: PropTypes.string,

    // setFieldValue(value: string);

    setFieldValue: PropTypes.func.isRequired,

}

IE提供的字段

BaseField：

其props如下

BaseField.propTypes = {

    text: PropTypes.string.isRequired,

    fieldValue: PropTypes.string,

    // setFieldValue(value: string);

    setFieldValue: PropTypes.func.isRequired,

}

BaseCustomizeField（抽象class，需要继承）：

考虑到有些组件需要复杂的数据结构所以提供了基本自定义字段，其代码如下

import React from 'react';

import PropTypes from 'prop-types';

export default class BaseCustomizeField extends React.Component{

    constructor(props){

        super(props);

        this.fieldValue = this.getFieldValue();

}

    // 在组件初始化时调用

    getFieldValue() {

        if(!this.props.fieldValue){

            return {};

        }

        let fieldValue;

        try{

            fieldValue = JSON.parse(this.props.fieldValue);

        }

        catch(e){

            return {};

        }

        return fieldValue;

}

    // 设置字段的值

    // 推荐在失去焦点时调用该函数

    setFieldValue() {

        if(!this.fieldValue){

            return;

        }

        this.props.setFieldValue(JSON.stringify(this.fieldValue));

    }

}

BaseCustomizeField.propTypes = {

    fieldValue: PropTypes.string,

    setFieldValue: PropTypes.func.isRequired,

    text: PropTypes.string,

}

继承示例：

import React from "react";

import { BaseCustomizeField } from '../BaseComponent'

class CustomizeField extends BaseCustomizeField {

    constructor(props) {

        super(props)

}

    render() {

        // fieldValue = {

        //     type: "",

        //     value: ""

        // };

        return (

            <div>

                <input value={this.fieldValue.type} type="text" className="form-control" placeholder={`请输入类型`}

                    onChange={

                        (event) => {

                            // 改变字段值

                            this.fieldValue.type = event.target.value;

                            this.setState({});

                        }

                    }

                    // 失去焦点时设置字段值

                    onBlur={() => { this.setFieldValue() }}

                />

                <input value={this.fieldValue.value} type="text" className="form-control" placeholder={`请输入占比（0~100）`}

                    onChange={

                        (event) => {

                            this.fieldValue.value = event.target.value;

                            this.setState({});

                        }

                    }

                    onBlur={() => { this.setFieldValue() }}

                />

            </div>

        );

    }

}

export default CustomizeField